MODULE 2

FINITE STATE MACHINES WITH OUTPUT

MOORE MACHINE AND MEALY MACHINE

- These machines can be described by **($Q, \Sigma, \delta, q_0, \Delta, \lambda$)**

Q – Finite set of states

Σ – Input alphabet

δ - Transition function ($Q \times \Sigma \rightarrow Q$)

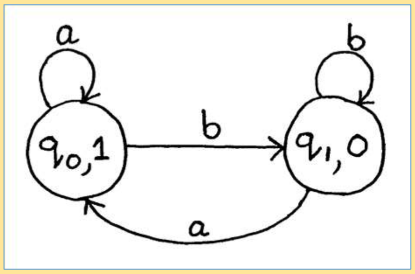
q_0 - Initial state

Δ - Output alphabet

λ – Output function

- The only difference between Moore and Mealy is in **λ**

MOORE MACHINE



$\lambda : Q \rightarrow \Delta$

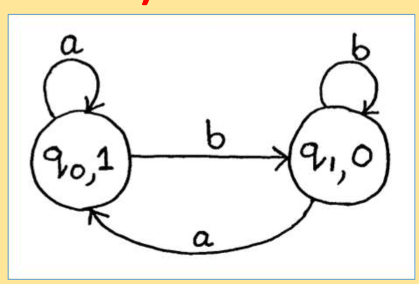
- Here for every state an output is associated
- Δ is a symbol which will be outputted by the machine.
- The state q_0 produce an output 1
- The state q_1 produce an output 0

MOORE MACHINE (WORKING)

- If we give input ab

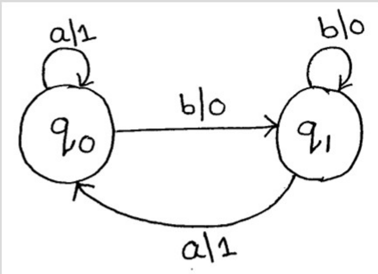
$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1$$

1 1 0



- Without seeing anything, q_0 will produce some output
- On seeing the input **ab** , this moore machine produce the output **110**
- In general, if the string of length **n** is input, then output produced is string of **n+1** length.

MEALY MACHINE



λ : Q×Σ→Δ

$(q_0, a) \rightarrow 1$

$(q_0, b) \rightarrow 0$

$(q_1, b) \rightarrow 0$

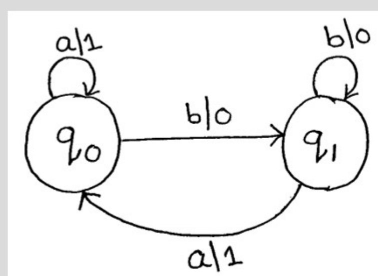
$(q_1, a) \rightarrow 1$

- For a state and a given input , there will be some output
- For state q_0 , if input is a, then output is 1

MEALY MACHINE (WORKING)

- Let the input is ab

$$q_0 \xrightarrow[1]{a} q_0 \xrightarrow[0]{b} q_1$$



- The output is associated with input
- If we give n bit input, the output will be n bit

MYHILL – NERODE THEOREM

- It implies that there is a unique minimal DFA with minimum number of states
- Minimization of DFA - DFA minimization stands for converting a given DFA to its equivalent DFA with minimum number of states.

  Here

  Input – DFA

  Output – Minimized DFA
- Table filling method is also called, Myhill - Nerode theorem.

Prepared By Mr.EBIN PM, AP, IESCE

EDULINE 7

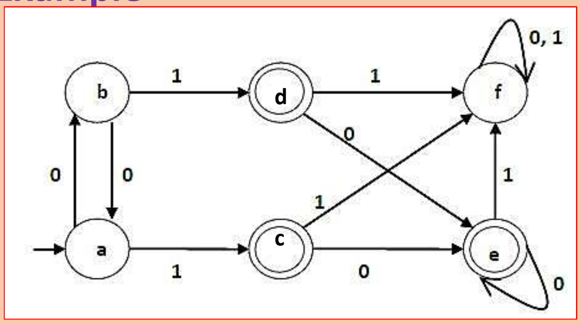
STEP

1. Draw a table for all pairs of states (P, Q)
2. Mark all pairs where P Є F and Q ∉ F
3. If there are many unmarked pairs (P, Q) such that [δ(P,x), δ(Q,x)] is marked , then mark [P, Q] (where x is an input symbol) . Repeat this until no more markings can be made.
4. Combine all the unmarked pairs and make them a single state in the minimized DFA.

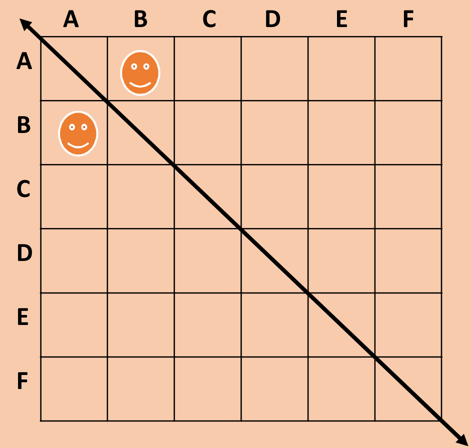
Prepared By Mr. EBIN PM, AP, IESCE

EDULINE 8

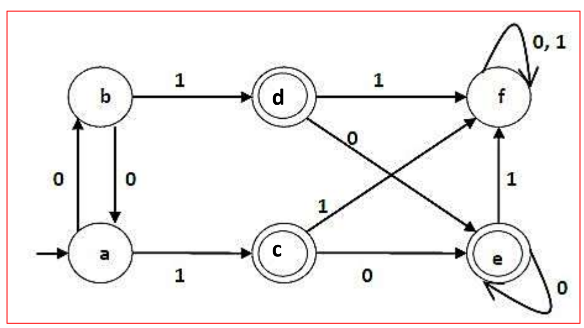
Example



Step 1: Draw a table for all pairs of states (P, Q)

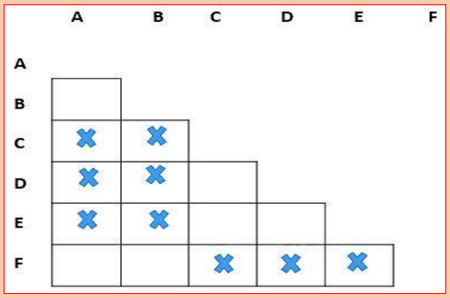


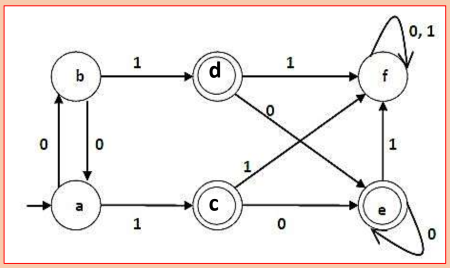
	A	B	C	D	E	F
A						
B						
C	X	X				
D	X	X				
E	X	X				
F			X	X	X	



Step 2: Mark all pairs where P Є F and Q ∉ F (One state should be final and other should be non final state)

Step 3



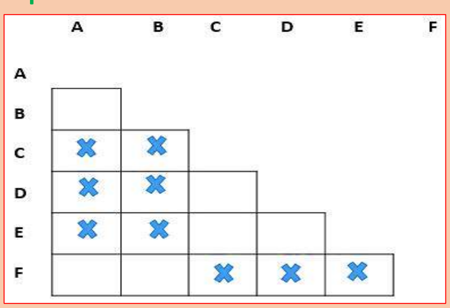


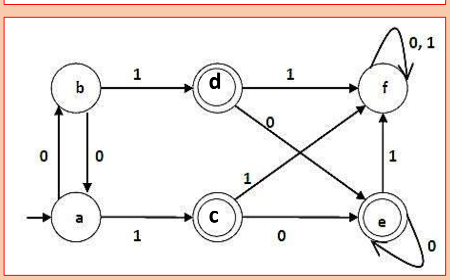
(B,A) - δ (B, 0) - A
δ (A, 0) – B unmarked
δ (B, 1) – D
δ (A, 1) – C unmarked

(D,C) - δ (D, 0) –E
δ (C, 0) – E no such column
δ (D, 1) – F
δ (C, 1) – F no such column

Prepared By Mr. EBIN PM, AP, IESCE EDULINE 11

Step 3 continuation





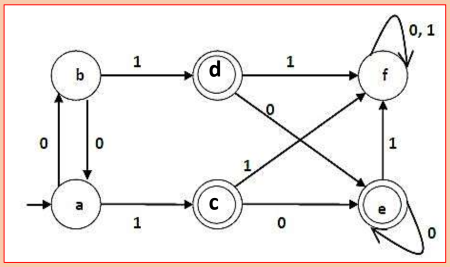
(E,C) - δ (E, 0) - E
δ (C, 0) – E no such column
δ (E, 1) – F
δ (C, 1) – F no such column

(E,D) - δ (E, 0) –E
δ (D, 0) – E no such column
δ (E, 1) – F
δ (D, 1) – F no such column

Prepared By Mr. EBIN PM, AP, IESCE EDULINE 12

Step 3 continuation

	A	B	C	D	E	F
A						
B						
C	✖	✖				
D	✖	✖				
E	✖	✖				
F			✖	✖	✖	



(F, A) - δ (F, 0) - F
δ (A, 0) – B unmarked
δ (F, 1) – F
δ (A, 1) – C marked. So we should mark FA

(F, B) - δ (F, 0) –F
δ (B, 0) –A FA is marked .so we should mark FB also.

Prepared By Mr. EBIN PM, AP, IESCE EDULINE

Final table

	A	B	C	D	E	F
A						
B						
C	✖	✖				
D	✖	✖				
E	✖	✖				
F	✖	✖	✖	✖	✖	

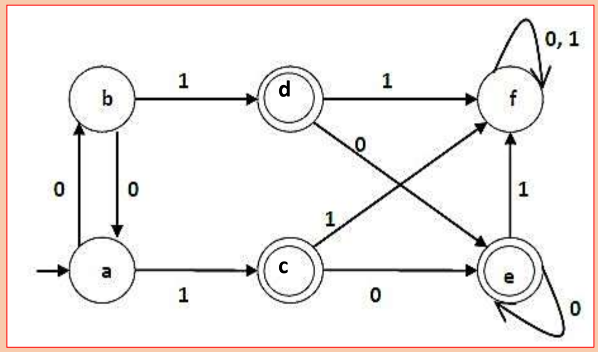
Prepared By Mr. EBIN PM, AP, IESCE EDULINE

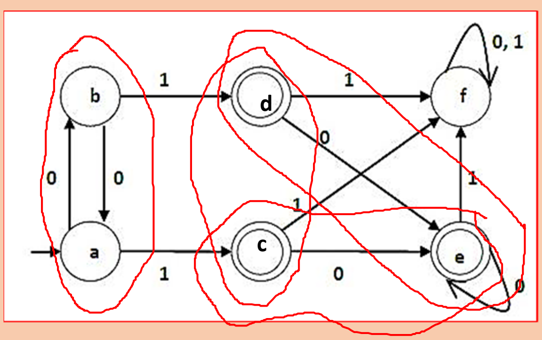
Step 4

	A	B	C	D	E	F
A						
B						
C	X	X				
D	X	X				
E	X	X				
F	X	X	X	X	X	

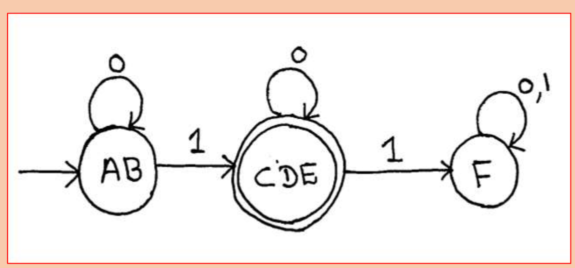
Combine the unmarked pairs

(A,B), (D,C), (E,C), (E,D)





Minimal DFA



DFA MINIMIZATION USING EQUIVALENCE THEOREM

- Suppose we have 2 states (P, Q)
- We can say that P and Q are equivalent , when

$$\delta(P, w) \in F \Rightarrow \delta(Q, w) \in F$$
$$\delta(P, w) \notin F \Rightarrow \delta(Q, w) \notin F$$

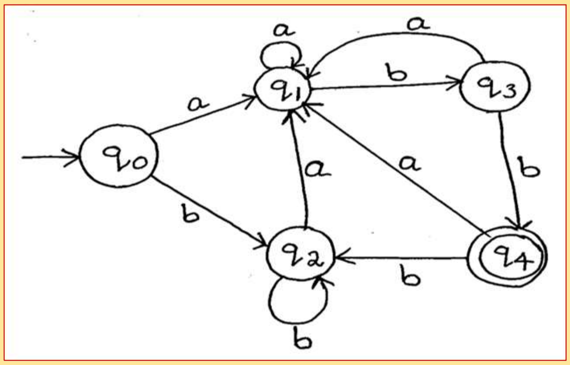
- If the above condition is satisfied, we can combine the states P and Q in to one state.
- The above condition can be used for combine the two states in to a single state.

- If $|w|=0$, then states P and Q are called 0 equivalent
- If $|w|=1$, then states P and Q are called 1 equivalent
- If $|w|=2$, then states P and Q are called 2 equivalent

In general,

- If $|w|=n$, then states P and Q are called n equivalent

Q: Minimize the following DFA



State Transition Table

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	$^*q_4$
$^*q_4$	q_1	q_2

- Here the start state is q_0 and final state is q_4.
- Step 1 - Identify the unreachable states (The states which are not reachable from initial state) . If such state exist, delete it.
- Step 2 – Draw state transition table

Prepared By Mr. EBIN PM, AP, IESCE EDULINE

- **Step 3 -** Find 0 equivalent states (ie, separate non final and final states)

  $[q_0, q_1, q_2, q_3]$ $[q_4]$
- Find 1 equivalent states.
- Check 1 equivalent of (q_0,q_1)
- Check 1 equivalent of (q_0,q_2)
- Check 1 equivalent of (q_2,q_3)

1 equivalent states

q_0 q_1 q_2 q_3 q_4

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	$^*q_4$
$^*q_4$	q_1	q_2

Prepared By Mr. EBIN PM, AP, IESCE EDULINE

➢Find 2 equivalent states

- 2 equivalent state is find using 1 equivalent states. ie,

[q_0, q_1, q_2] [q_3] [q_4]

- Check 2 equivalent of (q_0,q_1)
- Check 2 equivalent of (q_0,q_2)

2 equivalent states

q_0 q_2 q_1 q_3 q_4

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	$*q_4$
$*q_4$	q_1	q_2

Prepared By Mr. EBIN PM, AP, IESCE EDULINE

➢Find 3 equivalent states

- 3 equivalent state can be find using 2 equivalent states. ie,

[q_0,q_2] [q_1] [q_3] [q_4]

- Check 3 equivalent of (q_0,q_2)

3 equivalent states

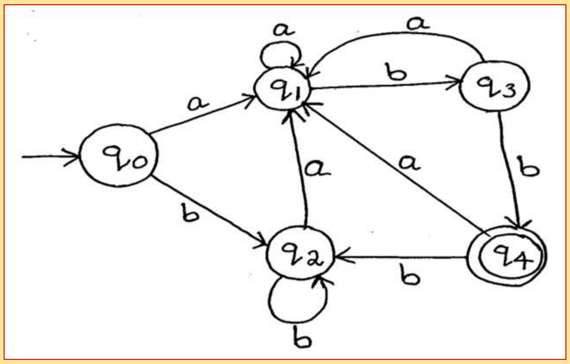
[q_0,q_2] [q_1] [q_3] [q_4]

ie, No further division is possible

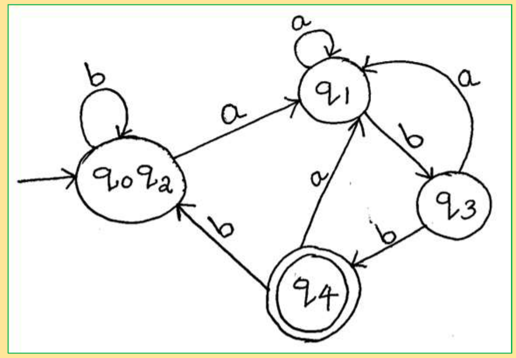
	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	$*q_4$
$*q_4$	q_1	q_2

Prepared By Mr. EBIN PM, AP, IESCE EDULINE

➢Step 4- Draw minimal DFA using the states



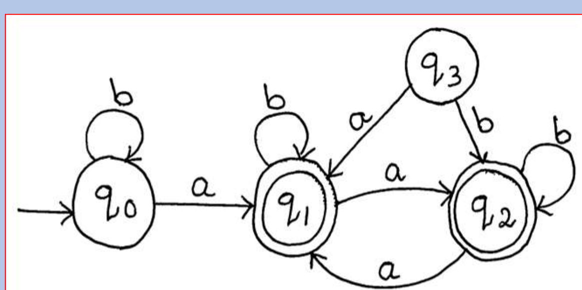
(This is our question)

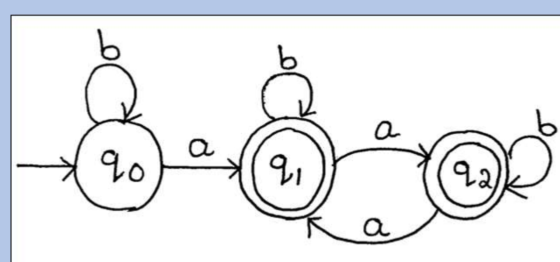


(minimized DFA - Answer)

Prepared By Mr. EBIN PM, AP, IESCE EDULINE 23

Q: Minimize the following DFA using equivalence theorem





➢q_3 is not reachable from starting state. Then we should delete q_3

Prepared By Mr. EBIN PM, AP, IESCE EDULINE 24

- Find 0 equivalent states

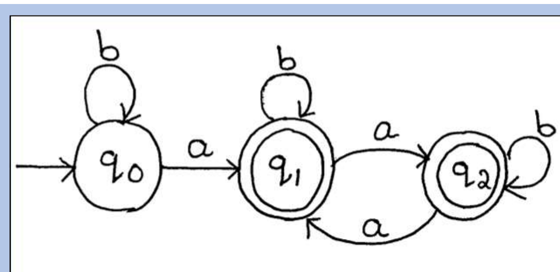
  [q_0] [q_1 q_2]

- Find 1 equivalent states
- Check 1 equivalent of (q_1 q_2)

1 equivalent states

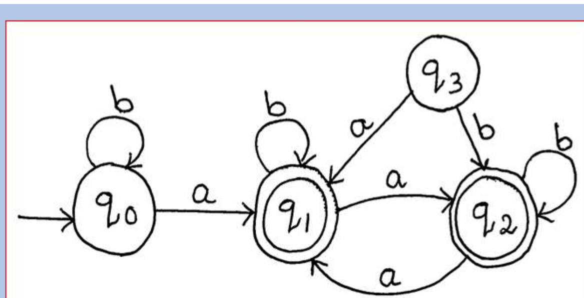
[q_1 q_2] [q_0]

No further division is possible

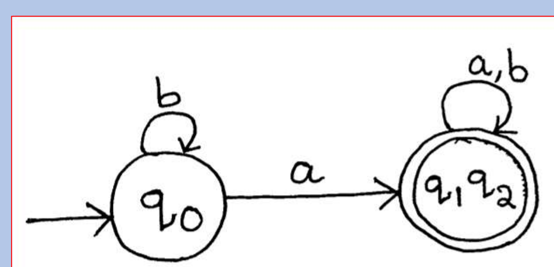


	a	b
→q_0	*q_1	q_0
*q_1	*q_2	*q_1
*q_2	*q_1	*q_2

Prepared By Mr. EBIN PM, AP, IESCE EDULINE 25



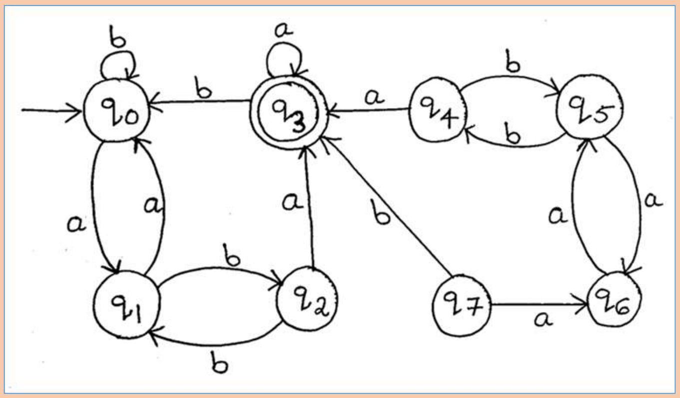
Given DFA –Question

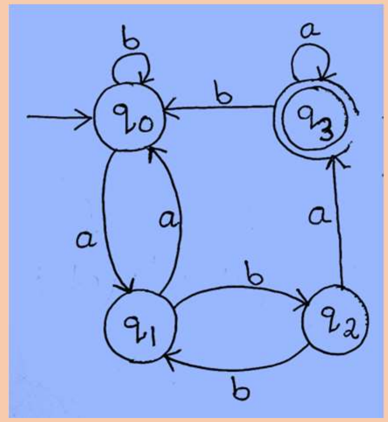


Minimized DFA - Answer

Prepared By Mr. EBIN PM, AP, IESCE EDULINE 26

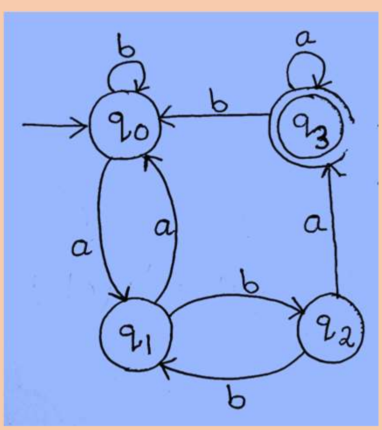
❖ Minimization of DFA (Example 3)





- Step1 - Identify the unreachable states
- Step 2- Draw state transition table

- Step 2- Draw state transition table



	a	b
→q0	q1	q0
q1	q0	q2
q2	*q3	q1
*q3	*q3	q0

➢Step 3 - Find equivalent states

0 equivalent

[q_0 q_1 q_2]   [q_3]

1 equivalent

- Check 1 equivalent of (q_0 q_1)
- Check 1 equivalent of (q_0 q_2)

	a	b
→q_0	q_1	q_0
q_1	q_0	q_2
q_2	*q_3	q_1
*q_3	*q_3	q_0

1 equivalent states

q_0 q_1 q_2 q_3

Prepared By Mr. EBIN PM, AP, IESCE   EDULINE

2 equivalent

- To find 2 equivalent states, we use 1 equivalent states. Ie,

[q_0 q_1]   [q_2]   [q_3]

- Check 2 equivalent of (q_0 q_1)

	a	b
→q_0	q_1	q_0
q_1	q_0	q_2
q_2	*q_3	q_1
*q_3	*q_3	q_0

2 equivalent states

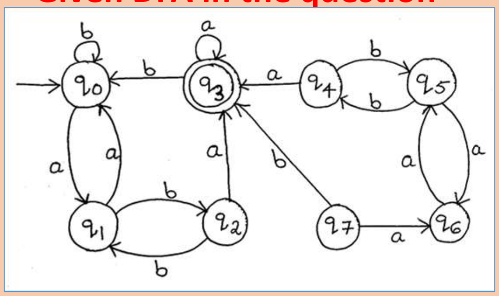
q_0 q_1 q_2 q_3

So, 2 equivalent states are

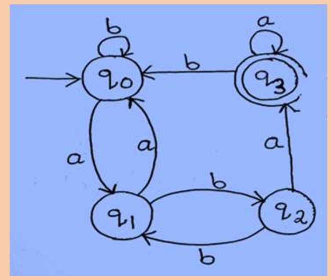
[q_0]   [q_1]   [q_2]   [q_3]

Prepared By Mr. EBIN PM, AP, IESCE   EDULINE

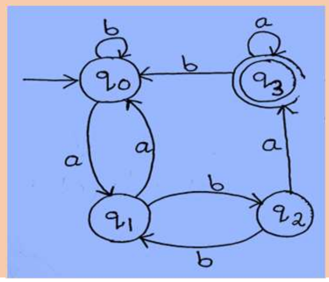
Given DFA in the question



After removing unreachable states



➢Minimized DFA, which is also Same as the above



GRAMMAR

➢A grammar describes how to form strings from a language's alphabet that are valid according to the language's syntax.

➢A grammar is usually thought of as a language generator.

Formal Definition

- A Grammar is a 4-tuple such that **G = (V , T , P , S)** where

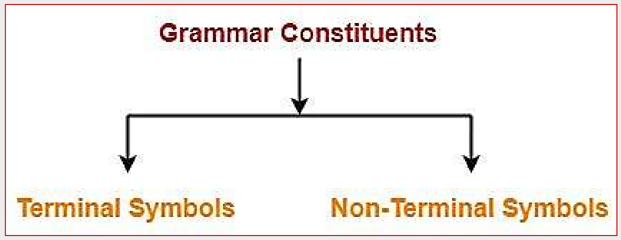
  V = Finite non-empty set of non-terminal symbols (Variables)

  T = Finite set of terminal symbols

  P = Finite non-empty set of production rules

  S = Start symbol

➢A Grammar is mainly composed of two basic elements



- Terminal Symbols - are denoted by using small case letters such as a, b, c etc.
- Non-Terminal Symbols - are denoted by using capital letters such as A, B, C etc. It is also called variables.

➢Example 1 of Grammar

- Consider a grammar G = (V , T , P , S) where-

V = { S } // Set of Non-Terminal symbols

T = { a , b } // Set of Terminal symbols

P = { S → aSbS , S → bSaS , S → ε } // Set of production rules

S = { S } // Start symbol

This grammar generates the strings having equal number of a's and b's

➢**Example 2 of Grammar**

Consider a grammar G = (V , T , P , S) where-

V = { S , A , B } // Set of Non-Terminal symbols

T = { a , b } // Set of Terminal symbols

P = { S → ABa , A → BB , B → ab , AA → b } // Set of production rules

S = { S } // Start symbol

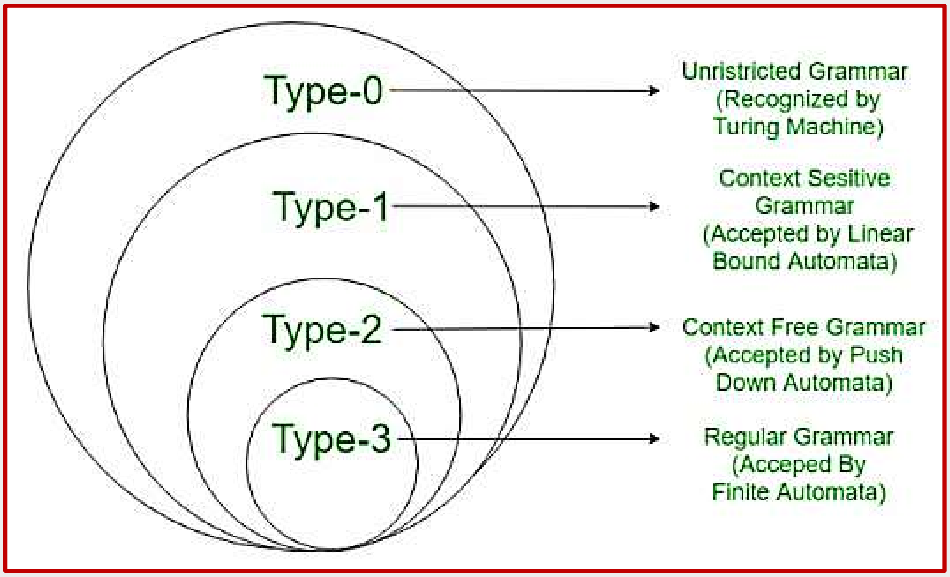
❖ Chomsky Hierarchy

➢Noam Chomsky gave a mathematical model of grammar in 1956 which is effective for writing computer languages.

➢According to Chomsky hierarchy, grammars are divided of 4 types:

- ❖ **Type 0** known as **unrestricted grammar**
- ❖ **Type 1** known as **context sensitive grammar**
- ❖ **Type 2** known as **context free grammar**
- ❖ **Type 3** Known as **Regular Grammar**

❖ Chomsky Hierarchy



Prepared By Mr. EBIN PM, AP, IESCE EDULINE 17

❖ Type 3: Regular Grammar

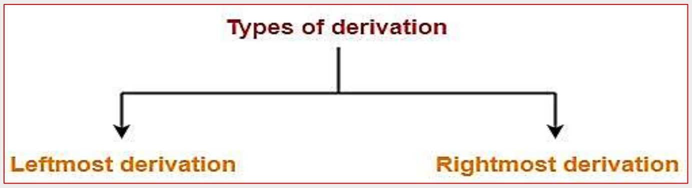
- Regular grammars generate regular languages.
- These languages are exactly all languages that can be accepted by a finite state automaton.
- Type 3 is most restricted form of grammar.
- Regular grammar contains the production of the form $\alpha \to \beta$ where $|\alpha| \leq |\beta|$, $\alpha \in v$ and β has the form aB or a.

  Eg : $S \to aS$, $S \to b$

Prepared By Mr. EBIN PM, AP, IESCE EDULINE 18

❖ Derivations

- The process of deriving a string is called as **derivation**.
- The geometrical representation of a derivation is called as a **parse tree** or **derivation tree.**



- **Leftmost Derivation** – It is the process of deriving a string by expanding the leftmost non-terminal at each step
- **Rightmost Derivation** – It is the process of deriving a string by expanding the rightmost non-terminal at each step

❖ Leftmost Derivation

- Find Leftmost derivation of the Following Grammar ?

$S \rightarrow AB \mid \varepsilon$

$A \rightarrow aB$

$B \rightarrow Sb$

Derive the string abb

$S \rightarrow AB$
$\rightarrow aBB$
$\rightarrow aSbB$
$\rightarrow a\varepsilon bB$
$\rightarrow abB$
$\rightarrow abSb$
$\rightarrow ab\varepsilon b$
$\rightarrow abb$

❖Rightmost Derivation

➢Find Rightmost derivation of the Following Grammar ?

$S \rightarrow AB \mid \varepsilon$

$A \rightarrow aB$

$B \rightarrow Sb$

Derive the string abb

$S \rightarrow AB$
$\rightarrow ASb$
$\rightarrow A\varepsilon b$
$\rightarrow aBb$
$\rightarrow aSbb$
$\rightarrow a\varepsilon bb$
$\rightarrow abb$

Prepared By Mr. EBIN PM, AP, IESCE EDULINE 21

REGULAR EXPRESSIONS

- ➢The language accepted by finite automata can be easily described by simple expressions called Regular Expressions.
- ➢It is the most effective way to represent any language.
- ➢The languages accepted by some regular expression are referred to as Regular languages.
- ➢A regular expression can also be described as a sequence of pattern that defines a string.
- ➢Regular expressions are used to match character combinations in strings.

Prepared By Mr. EBIN PM, AP, IESCE EDULINE 22

- Regular Expressions are used to denote regular languages. An expression is regular if
  - ϕ is a regular expression for regular language ϕ.
  - ε is a regular expression for regular language {ε}
  - If a ∈ Σ (Σ represents the input alphabet), a is regular expression with language {a}.
  - If a and b are regular expression, a + b is also a regular expression with language {a,b}.
  - If a and b are regular expression, ab (concatenation of a and b) is also regular.
  - If a is regular expression, a* (0 or more times a) is also regular.

- Regular Languages : A language is regular if it can be expressed in terms of regular expression.
- In a regular expression, x^* means zero or more occurrence of x. It can generate {ε, x, xx, xxx, xxxx, ...}
- In a regular expression, x^+ means one or more occurrence of x. It can generate {x, xx, xxx, xxxx, ...}

Example 1:

Write the regular expression for the language accepting all combinations of a's, over the set ∑ = {a}

Solution:

- All combinations of a's means a may be zero, single, double and so on.
- If a is appearing zero times, that means a null string. That is we expect the set of {ε, a, aa, aaa, ...}.
- So we give a regular expression for this as:

$$RE = a^*$$

Example 2:

Write the regular expression for the language accepting all combinations of a's except the null string, over the set ∑ = {a}

Solution:

- The regular expression has to be built for the language

  L = {a, aa, aaa, ...}

- This set indicates that there is no null string. So we can denote regular expression as:

$$RE = a^+$$

Example 3:

Write the regular expression for the language accepting all the string containing any number of a's and b's.

Solution:

The regular expression will be:

RE = **(a + b)***

- This will give the set as L = {ε, a, aa, b, bb, ab, ba, aba, bab, ...}, any combination of a and b.
- The (a + b)* shows any combination with a and b even a null string.

Example 4:

Write the regular expression for the language accepting all the string which are starting with 1 and ending with 0, over ∑ = {0, 1}

Solution:

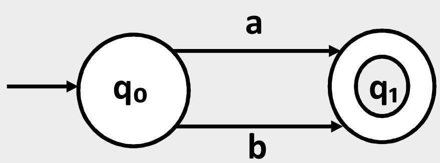
- In a regular expression, the first symbol should be 1, and the last symbol should be 0.

RE = 1 (0+1)* 0

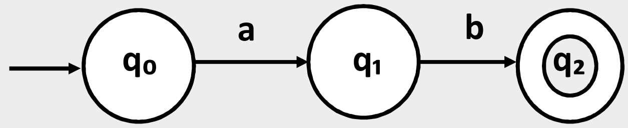
Conversion of RE to FA

Rules:

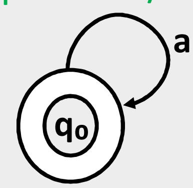
1. a+b (a or b, a|b)



2. (ab)



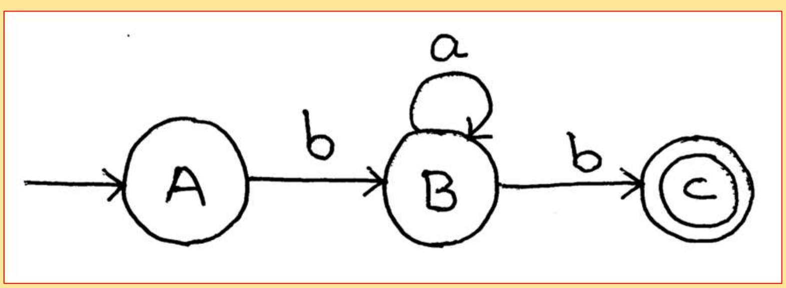
3. a^* (Repetition / a closure)



Example: 1

Convert the regular expression ba^*b to Finite Automata

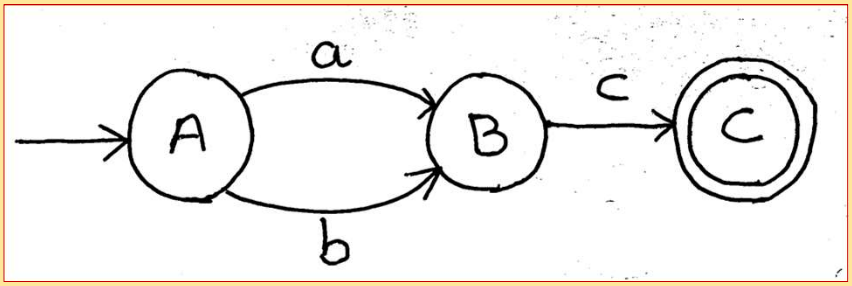
L= {bb, bab, baab,…….}



Example: 2

Convert the regular expression **(a+b)c** to Finite Automata

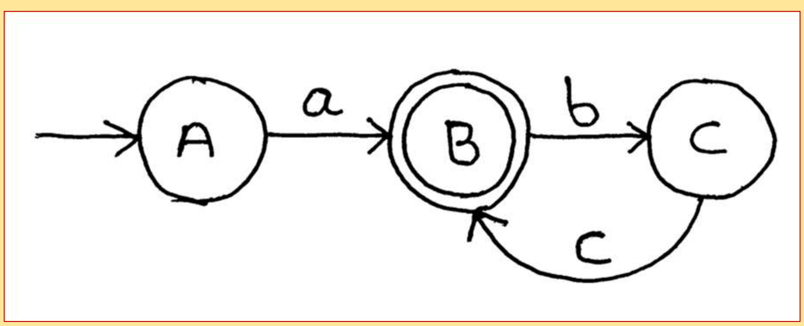
L= {ac, bc}



Example: 3

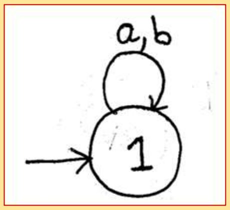
Convert the regular expression $a(bc)^*$ to Finite Automata

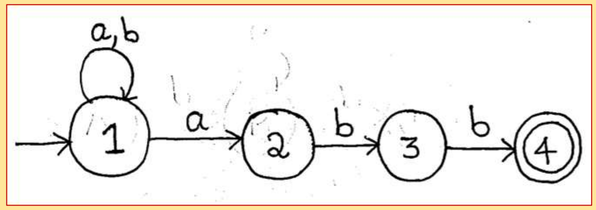
L= {a, abc, abcbc, abcbcbc, ……..}

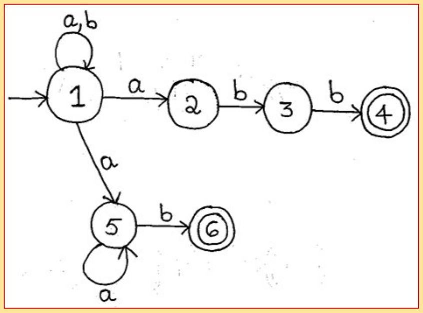


Example: 3

Convert the regular expression $(a|b)^*$ (abb|a^+b) to Finite Automata





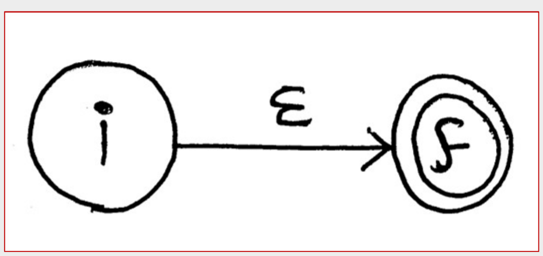


REGULAR EXPRESSION TO NFA

➢ The following are the rules to convert a RE to NFA. They are called Thompson's Rule

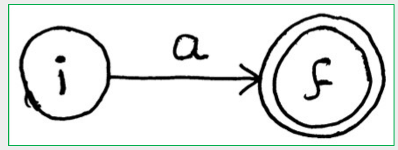
Rule: 1

- Let i be the initial state and f be the final state. The NFA for the regular expression (RE) that accept null string (ε) is given by



Rule: 2

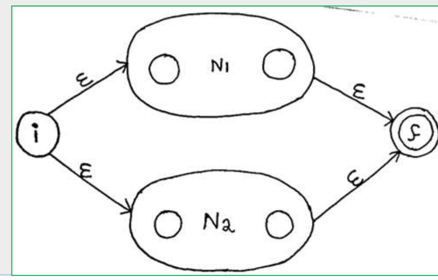
- The NFA for the regular expression (RE) that accept an input symbol **a** is given by



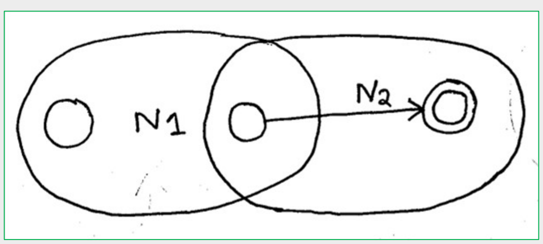
Rule : 3

- If N1 and N2 are NFA for the regular Expression R1 and R2,then

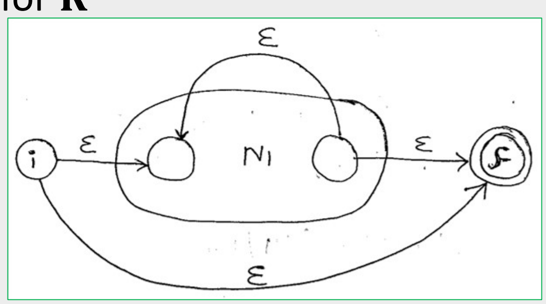
(i) NFA for R1+R2 (R1|R2)



(i) NFA for R1R2



(ii) NFA for $\mathbf{R}^*$



Example : 1

Construct NFA for the RE **(a|b)*a**

W can split the RE like

R1 = a

R2 = b

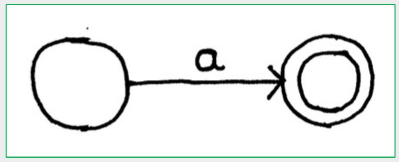
R3 = R1 | R2

R4 = R3*

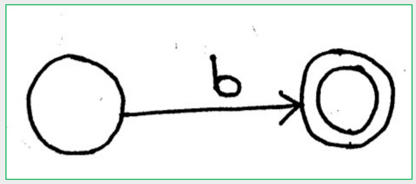
R5 = R4.R1

We can construct NFA for each one

for a



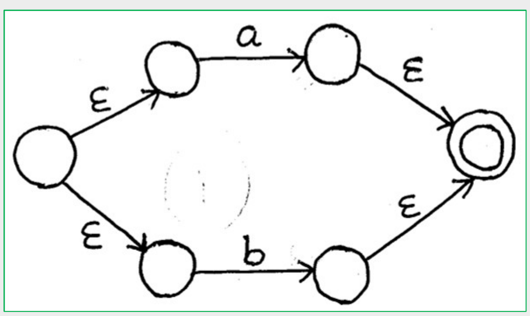
for b



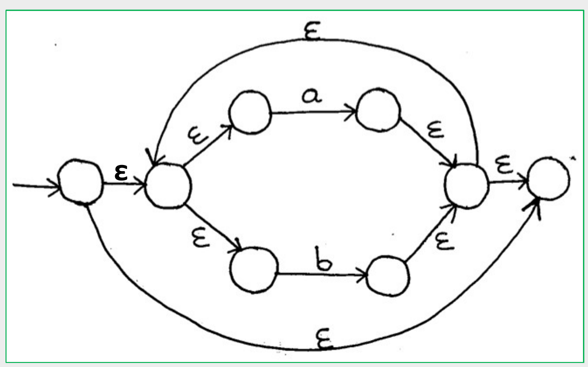
Prepared By Mr. EBIN PM, AP, IESCE

EDULINE

for **a|b**



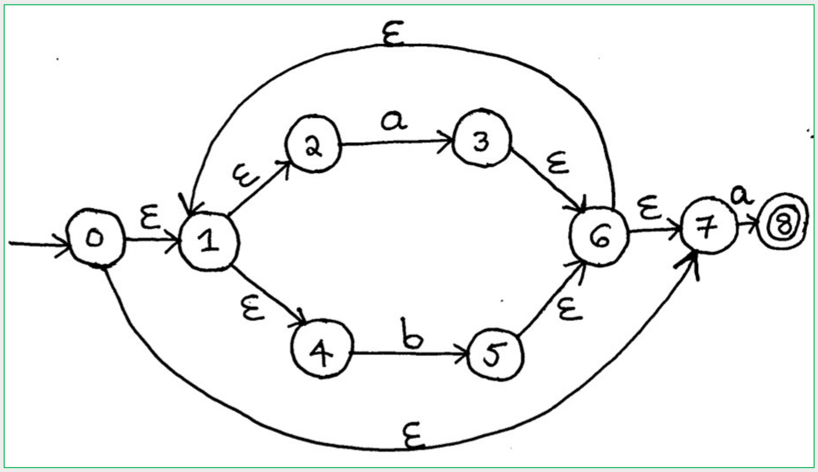
for **(a|b)***



Prepared By Mr. EBIN PM, AP, IESCE

EDULINE

for **(a|b)*a**



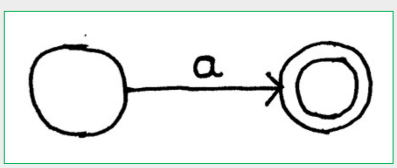
Prepared By Mr. EBIN PM, AP, IESCE

EDULINE

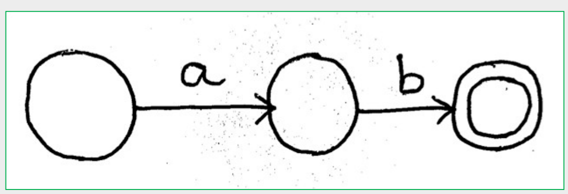
Example 2

- Draw NFA to represent the regular expression **ab|(a|b)a**

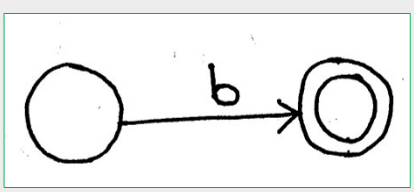
for a



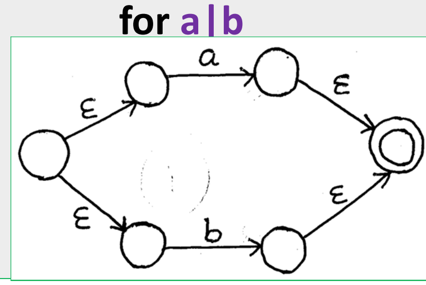
for ab



for b



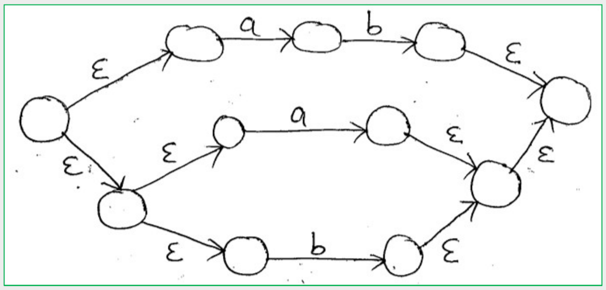
for a|b



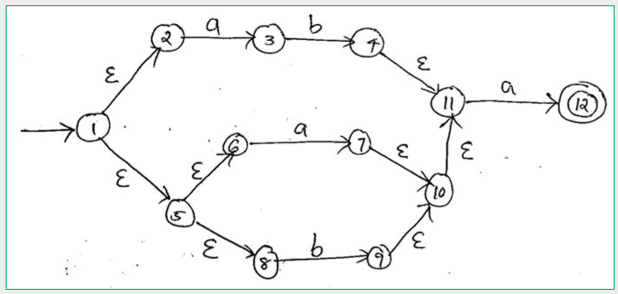
Prepared By Mr. EBIN PM, AP, IESCE

EDULINE

for ab|(a|b)



for ab|(a|b)a

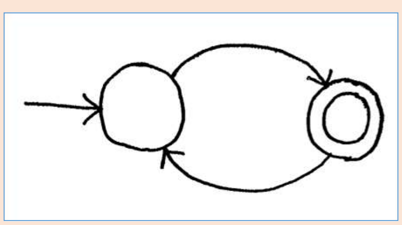


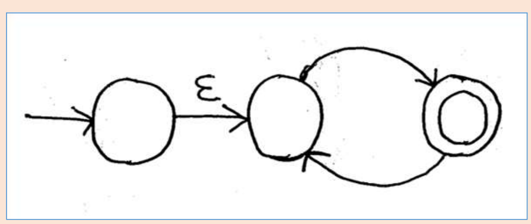
Conversion of FA to RE

- Here we are using state elimination method

Rule 1

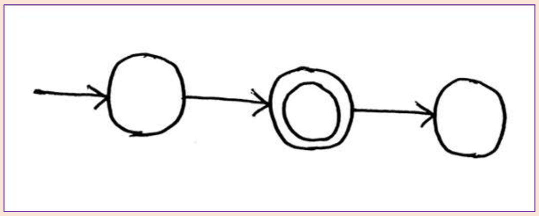
- Initial state should not have any incoming edge from other state. If incoming edge is present, then create new initial state

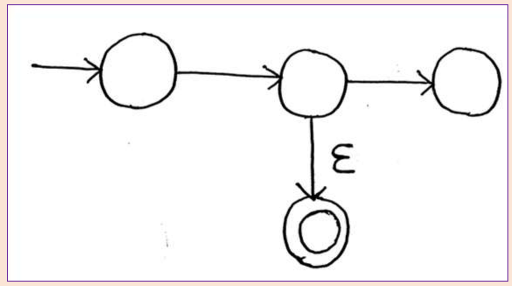




Rule 2a

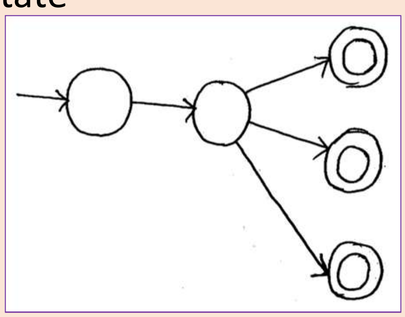
➢Final state should not have any outgoing edge. If outgoing edge is present, then create new final state

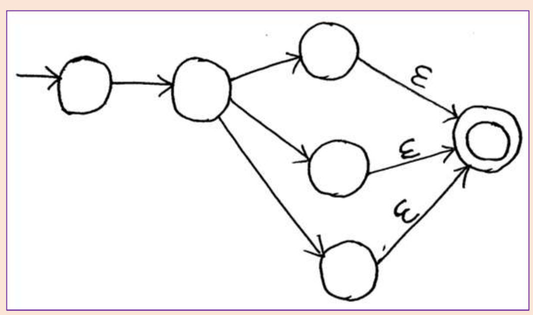




Rule 2b

➢If more than one Final state is present, then convert it into one state

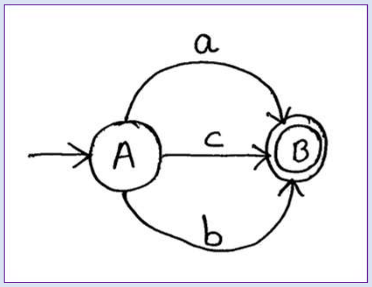




Rule 3

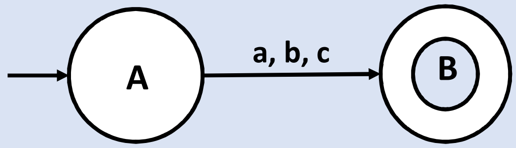
➢Eliminate the state one by one other than the initial & final state

Example 1



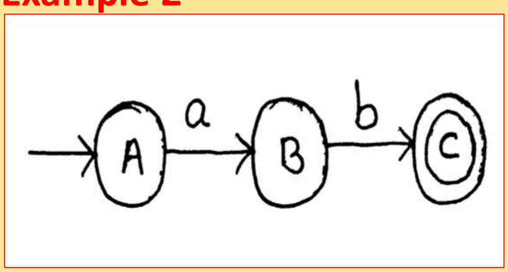
No incoming edge from other state to initial state

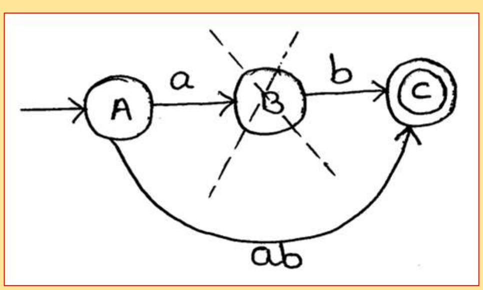
No outgoing edges from final state.



RE = **(a+b+c)**

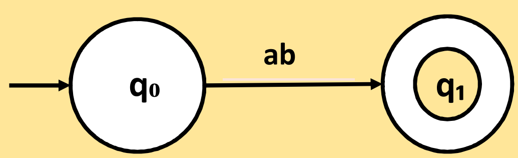
Example 2





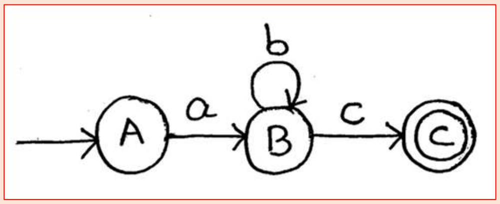
No incoming edge from other state to initial state

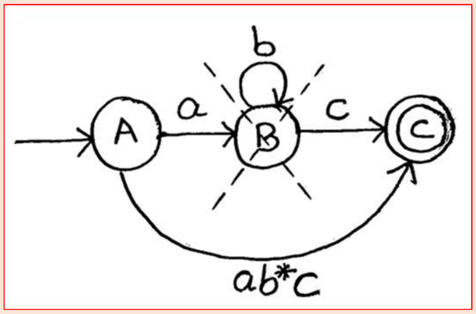
No outgoing edges from final state.



RE = **ab**

Example 3





No incoming edge from other state to initial state

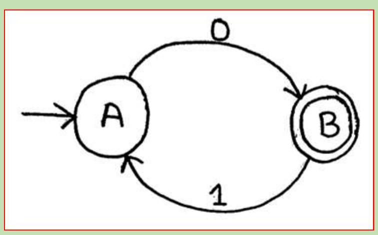
No outgoing edges from final state.

RE = **ab*c**

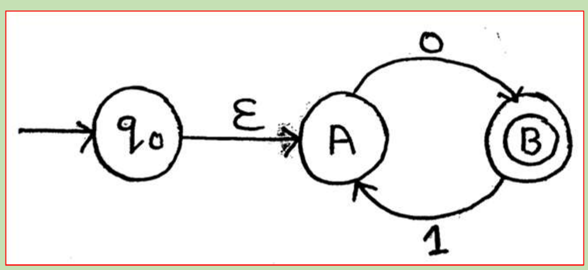
Prepared By Mr. EBIN PM, AP, IESCE EDULINE

Example 4

Step - 1

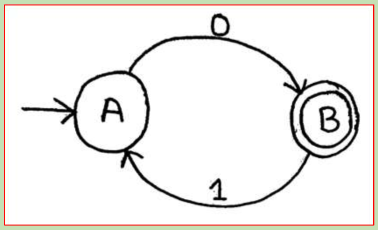


Incoming edge is present from other state to initial state . So , create new initial state

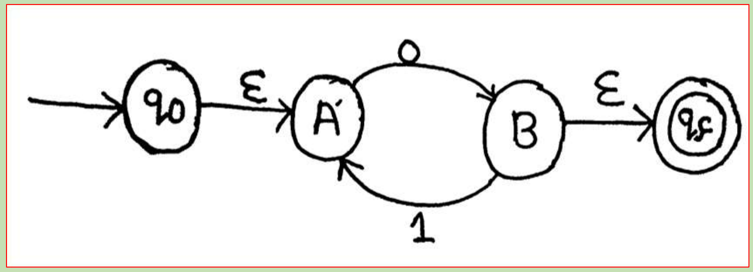


Prepared By Mr. EBIN PM, AP, IESCE EDULINE

Step - 2

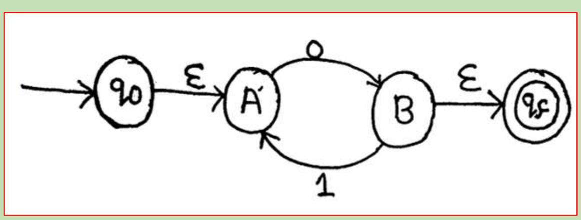


Outgoing edge is present from final state
So , create new final state

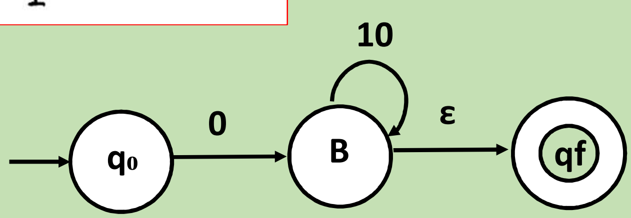


Step - 3

Delete all intermediate states one by one

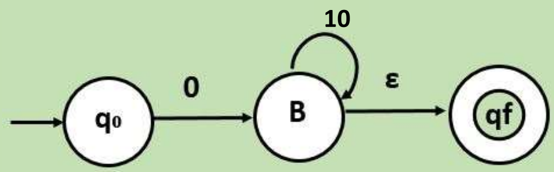


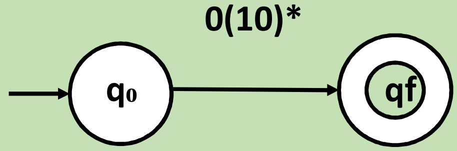
Eliminate the state A



Step - 3

Eliminate the state B





RE = **0(10)***